Computing Behavioural Relations, Logically

Rance Cleaveland*
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
USA

Bernhard Steffen
Lehrstuhl für Informatik II
RWTH Aachen
Ahornstraße 55
W-5100 Aachen
GERMANY

Abstract

This paper develops a model-checking algorithm for a fragment of the modal mu-calculus and shows how it may be applied to the efficient computation of behavioral relations between processes. The algorithm's complexity is proportional to the product of the size of the process and the size of the formula, and thus improves on the best existing algorithm for such a fixed point logic. The method for computing preorders that the model checker induces is also more efficient than known algorithms.

1 Introduction

Equivalences, preorders and logical formulas have all proved to be useful in the specification and verification of processes. Equivalences enable implementations to be related to *complete* specifications (which are also processes) when the two exhibit the same behavior, while preorders permit implementations to be deemed correct when they provide "at least" the behavior stipulated by a *partial* process specification. Formulas, on the other hand, allow one to establish whether *specific properties* hold of implementations. In the case of finite-state processes, these techniques may be automated [BSV, CES, CPS1, CPS2, Fe, MSGS, RRSV].

In this paper we present a linear-time model checking algorithm for a variant of the modal mu-calculus [Ko, Sti, PS] and illustrate how it may be used to compute behavioral preorders efficiently. The latter result relies on the fact that the logic is expressive enough to characterize processes up to the prebisimulation preorder: a process is larger than another in the preorder if and only if the larger process satisfies the characteristic formula of the smaller [Ste, SI]. As characteristic formulas can be automatically constructed in time linear in the size of the argument processes, this yields a method for preorder checking that outperforms the known algorithms. Moreover, the prebisimulation preorder may be used as the basis for defining many behavioral preorders and equivalences on processes. Thus our approach covers these relations as well.

Our results may therefore be summarized as follows. We develop a model checker that:

*Research supported by National Science Foundation/DARPA Grant CCR-9014775

- has complexity that is linear in the size of the process and in the size of the formula. To our knowledge, the best algorithm that has been published for a similar fixed point logic is also linear in the size of the process, but quadratic in the size of the formula [AC].
- may be used to compute *preorders* between processes. The resulting *preorder checker*, which works by constructing characteristic formulas, is linear in the product of the sizes of the argument processes. This also improves on the known complexity results about preorder checking ([CPS1, CPS2]).

The remainder of the paper develops as follows. The next section presents our process model (transition systems) and logic. The section following then describes our model-checking algorithm, while Section 4 details the application of this algorithm to the computation of equivalences and preorders. The final section contains our conclusions and directions for future research.

2 Transition Systems and Modal Logic

We use *extended labeled transition systems* [Sti, Wa] to model processes. These may be formally defined as follows.

Definition 2.1 *An* extended labeled transition system $\mathcal{T}$ *is a quadruple* $\langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$, *where:*

- $\mathcal{S}$ *is a set of* states*;*
- Act *is a set of* actions*;*
- $\rightarrow \subseteq \mathcal{S} \times Act \times \mathcal{S}$ *is the* transition relation*; and*
- $\uparrow \subseteq \mathcal{S} \times Act$ *is the* divergence relation*.*

Intuitively, an extended labeled transition system encodes the operational behavior of a system. The set $\mathcal{S}$ represents the set of states the system may enter, and Act contains the set of actions the system may perform. The relation $\rightarrow$ describes the actions available to states and the state transitions that may result upon execution of the actions. The relation $\uparrow$ represents a notion of *partial definedness*: $\langle s, a \rangle \in \uparrow$ means that the behavior of s on action a may be *undefined*, or *divergent*.

In the remainder of the paper we write $s \xrightarrow{a} s'$ and $s \uparrow a$ in place of $\langle s, a, s' \rangle \in \rightarrow$ and $\langle s, a \rangle \in \uparrow$. We also use $s \downarrow a$ in place of $\neg(s \uparrow a)$ and $s \uparrow$ (resp. $s \downarrow$) to mean "for all $a \in Act$, $s \uparrow a$ (resp. $s \downarrow a$)", and we write $s \xrightarrow{a}$ when there is an s' such that $s \xrightarrow{a} s'$. If $s \xrightarrow{a} s'$ then we say that s' is an *a-derivative* of s.

Given an extended labeled transition system $\mathcal{T} = \langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$, we define processes as *rooted transition systems*, i.e. as pairs $(\mathcal{T}, s)$, where $s \in \mathcal{S}$ is a distinguished element, the "start state". If the transition system is obvious from the context, we omit reference to it when defining processes; in this case, processes will be identified with states. Finally, when $\mathcal{S}$ and Act are finite, we shall say that the extended labeled transition system is *finite-state*.

The logic we consider may be viewed as a variant of the modal mu-calculus [Ko], or the Hennessy-Milner Logic with recursion [La]. Let Var be a (countable) set of variables, $\mathcal{A}$ a set of atomic propositions, and Act a set of actions. In what follows, X will range over Var, A over $\mathcal{A}$, and a over Act. Then the syntax of *basic* formulas is given by the following grammar.

$$\Phi ::= A \mid X \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid \langle a \rangle \Phi \mid [a]\Phi$$

Formulas are interpreted with respect to a fixed extended labeled transition system $\langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$, a valuation $\mathcal{V} : \mathcal{A} \rightarrow 2^{\mathcal{S}}$, and an environment $e : Var \rightarrow 2^{\mathcal{S}}$.

$$
\begin{array}{rcl}
[\![A]\!]e & = & \mathcal{V}(A) \\
[\![X]\!]e & = & e(X) \\
[\![\Phi_1 \vee \Phi_2]\!]e & = & [\![\Phi_1]\!]e \cup [\![\Phi_2]\!]e \\
[\![\Phi_1 \wedge \Phi_2]\!]e & = & [\![\Phi_1]\!]e \cap [\![\Phi_2]\!]e \\
[\![\langle a \rangle \Phi]\!]e & = & \{\, s \mid \exists s'.\, s \xrightarrow{a} s' \wedge s' \in [\![\Phi]\!]e \,\} \\
[\![[a]\Phi]\!]e & = & \{\, s \mid s{\downarrow}a \wedge \forall s'.\, s \xrightarrow{a} s' \Rightarrow s' \in [\![\Phi]\!]e \,\}
\end{array}
$$

Figure 1: The semantics of basic formulas.

The formal semantics appears in Figure 1. It is given with respect to an extended labeled transition system $\langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$, a valuation $\mathcal{V}$ mapping atomic propositions to subsets of $\mathcal{S}$, and an environment e mapping variables to subsets of $\mathcal{S}$. Intuitively, the semantic function maps formulas to the sets of states for which the formula is "true". Accordingly, a state s satisfies $A \in \mathcal{A}$ if s is in the valuation of A, while s satisfies X if s is an element of the set bound to X in e. The propositional constructs are interpreted in the usual fashion: s satisfies $\Phi_1 \vee \Phi_2$ if it satisfies one of the Φ_i and it satisfies $\Phi_1 \wedge \Phi_2$ if it satisfies both of them. The constructs $\langle a \rangle$ and $[a]$ are *modal operators*; s satisfies $\langle a \rangle \Phi$ if it has an a-derivative satisfying Φ, while s satisfies $[a]\Phi$ if $s{\downarrow}a$ holds and each a-derivative of s satisfies Φ. It should be noted that $[a]\Phi$ is interpreted somewhat differently than in the classical modal mu-calculus; in fact, our logic may be seen as a fragment of an *intuitionistic* variant of the standard modal mu-calculus along the lines of [PS, Sti].

We may also define formulas using mutually recursive sets of equations. Consider a system of equations E

$$
\begin{array}{rcl}
X_1 & = & \Phi_1 \\
 & \vdots & \\
X_n & = & \Phi_n,
\end{array}
$$

where each Φ_i is a basic formula and each X_i is distinct. Given a fixed environment e, we may build a function $f_E : (2^{\mathcal{S}})^n \rightarrow (2^{\mathcal{S}})^n$ as follows. Let $S_i \subseteq 2^{\mathcal{S}}$ for $i = 1, \ldots, n$, and let $e_S = e[X_1 \mapsto S_1, \ldots, X_n \mapsto S_n]$ be the environment that results from e by updating the binding of X_i to S_i. Then

$$f_E(\langle S_1, \ldots, S_n \rangle) = \langle [\![\Phi_1]\!]e_S, \ldots, [\![\Phi_n]\!]e_S \rangle.$$

Now $(2^{\mathcal{S}})^n$ forms a complete lattice, where the ordering and the join and meet operations are the pointwise extensions of the set-theoretic inclusion $\subseteq$, union $\cup$ and intersection $\cap$, respectively. Moreover, for any E and environment e, f_E is monotonic with respect to $\subseteq$ and therefore, according to the Tarski fixed point theorem [Ta], has a *greatest* fixed point, νf_E. In general, νf_E has the following description.

$$\nu f_E \quad = \quad \bigcup \{\, \overline{S} \mid \overline{S} \subseteq f_E(\overline{S}) \,\}$$

In the case that the extended labeled transition system is finite-state, f_E is continuous, and νf_E also has an iterative characterization. Let

$$f_0 \quad = \quad \langle \mathcal{S}, \ldots, \mathcal{S} \rangle \text{ and}$$

$$f_{i+1} \quad = \quad f_E(f_i) \ \text{ for } i \geq 0.$$

Then $\nu f_E = \bigcap_{i=0}^{\infty} f_i$.

We now define what it means for a state in a transition system to satisfy a formula whose variables are "bound" by a set of equations. First, we say that a basic proposition Φ is *closed* with respect to a set of equations E if every variable in Φ appears on the left-hand side of some equation in E. We also refer to a set of equations E as closed if each right-hand side in E is closed with respect to E. Notice that for closed systems of equations E the maximal fixed point νf_E does not depend on the environment e used in the definition of f_E.

Now let $\nu f_E = \langle \nu_1, \ldots, \nu_n \rangle$ be the greatest fixed point for a closed system of equations E whose left-hand sides form the set $\{X_1, \ldots, X_n\}$. Moreover let Φ be a proposition being closed with respect to E and e an environment mapping X_i to ν_i. Then we write

$$s \models_E \Phi$$

to mean $s \in [\![\Phi]\!]e$.

In the remainder of this paper we restrict our attention to finite-state extended labeled transition systems.

3 Efficient Model Checking

In this section, we present an algorithm, **solve**, for computing νf_E, given a finite-state extended labeled transition system $\mathcal{T} = \langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$ and a closed set of equations E. The algorithm's complexity is $O(|\mathcal{T}| * |E|)$, where $|\mathcal{T}| = |\mathcal{S}| + |\rightarrow|$ and $|E|$ is the number of symbols in the equation system E.

Following [AC], we restrict our attention to equation sets E whose right-hand sides are *simple*, i.e. only have variables as nontrivial subterms and do not just consist of a variable. So $X_4 \vee X_3$ is simple, while $\langle a \rangle (X_4 \vee X_3)$ and X_4 are not. Any equation set E may be transformed in linear time into a simple equation set E' with at most a linear blow-up in size. Accordingly, **solve** has the same complexity for our full logic as it does for the simple sublogic.

3.1 Overview

As with the algorithm in [AC], **solve** is bit-vector-based. Each state in $\mathcal{S}$ will have a bit vector whose i^{th} entry indicates whether or not the state belongs to the set associated with X_i in the current stage of the analysis.

Initially, each component in every state's bit vector is set to *true*, with the following exceptions.

- The right-hand side of the corresponding equation is atomic, and the state does not satisfy the atomic proposition.
- The right-hand side of the corresponding equation is of the form $\langle a \rangle X_j$, and the state has no a-derivatives.
- The right-hand side of the corresponding equation is of the form $[a]X_j$, and the state diverges on a.

Notice that when a component in a state's bit vector is set to *false*, the state is effectively eliminated from the set associated with the proposition variable corresponding to the component.

The algorithm then iteratively examines the bit-vector annotations of each state, and, if necessary, sets components to *false* on the basis of components that have been previously set to false, until no further changes are required.

The key innovation in **solve** is that when a bit-vector component is set to *false*, the only components that are reinvestigated are those directly influenced by this change. This is in contrast to [AC], where the reinvestigation of components is less controlled; this accounts for their algorithm's higher complexity.

3.2 Data Structures

Assume that E is of the form $X_i = \Phi_i$, where i ranges between 1 and n. As in [AC], each state s will have the following fields associated with it.

- An array $X[1..n]$ of bits. Intuitively, $s.X[i]$ is true if s belongs to the set associated with proposition variable X_i. The array is initialized as described above.

- An array $C[1..n]$ of counters. If $X_i = X_j \vee X_k$ is an equation in E, then $s.C[i]$ records the number of disjuncts (0,1 or 2) of the right-hand side that are true for s. In this case, $s.C[i] = 2$ initially. If $X_i = \langle a \rangle X_j$ is in E then $s.C[i]$ records the number of a-derivatives of s that are in the set associated with X_j. In this case, $s.C[i]$ is initially set to the number of a-derivatives that s has. The counters are not used in the other cases.

- A field $s.A$ for every atomic proposition A that indicates whether s satisfies A or not. This is assumed to be given at the start of the algorithm.

In addition, the algorithm maintains two other data structures that allow one to determine efficiently which state/variable pairs must be reinvestigated. These structures enable the algorithm to run in time linear in the size of the equation system.

- A *list* M of state-variable pairs; $\langle s, X_i \rangle$ is in M if $s.X[i]$ has just been changed from *true* to *false*.

- An edge-labeled directed graph G with n vertices, one for each variable mentioned in E. The edges are defined as follows.

  - $X_i \xrightarrow{\vee} X_j$ if there is an X_k such that either $X_j = X_i \vee X_k$ or $X_j = X_k \vee X_i$ is an equation in E.
  - $X_i \xrightarrow{\wedge} X_j$ if there is an X_k such that either $X_j = X_i \wedge X_k$ or $X_j = X_k \wedge X_i$ is an equation in E.
  - $X_i \xrightarrow{\langle a \rangle} X_j$ if $X_j = \langle a \rangle X_i$ is in E.
  - $X_i \xrightarrow{[a]} X_j$ if $X_j = [a]X_i$ is in E.

  Intuitively, there is an edge from X_i to X_j if the set of states associated with X_i directly influences the set of states associated with X_j. This graph may be constructed in $O(|E|)$ time from E, and it contains no more than $2n$ edges (recall that n is the number of equation in E).

3.3 The Algorithm and its Complexity

The procedure **solve** computes νf_E in two stages.

- The *initialization* stage provides the data structures with their initial values; moreover, each pair $\langle s, X_i \rangle$ for which s.`X[i]` is set to false is added to M. Details of this procedure, called **initialize**, may be found in the appendix.
- The *update* stage successively deletes pairs $\langle s, X_i \rangle$ from the list M and processes them as follows:
  - For every X_j such that $X_i \xrightarrow{\vee} X_j$, the counter $s.C[j]$ is decremented by one. If $s.C[j]$ is now 0, then none of the disjuncts on the right-hand side of X_j are satisfied by s, and s must be removed from the set associated with X_j. Accordingly, $s.X[j]$ is set to *false* and the pair $\langle s, X_j \rangle$ is added to M.
  - For every X_j such that $X_i \xrightarrow{\wedge} X_j$, if $s.X[j]$ is *true* then the component $s.X[j]$ is set to *false* and the pair $\langle s, X_j \rangle$ is added to M.
  - For every X_j with $X_i \xrightarrow{\langle a \rangle} X_j$, each counter $C[j]$ for each s' such that $s' \xrightarrow{a} s$ is decremented by one, and if it becomes 0 (meaning that s' now has no a-derivatives satisfying X_i), then $s'.X[j]$ is set to *false* and $\langle s', X_j \rangle$ is added to M.
  - For every X_j with $X_i \xrightarrow{[a]} X_j$, each state s' such that $s' \xrightarrow{a} s$ has its $X[j]$-component examined, and if it is *true* then it is changed to *false* and $\langle s', X_j \rangle$ is added to M.

  Details of this procedure, called **update**, may be found in the appendix.

The procedure **solve** now consists of a call to **initialize** and a call to **update**. It always terminates, since the number of states is finite and for any state s and any i, the component $s.X[i]$ can be changed at most once, from *true* to *false*, during execution. Moreover, upon termination (i.e. when M is empty), the bit-vector annotations represent a fixed point of f_E that is in fact the greatest fixed point νf_E; this follows from the following observations.

- The initial bit-vector annotation "contains" νf_E.
- Processing a pair $\langle s', X_i \rangle$ transforms a bit-vector annotation that contains νf_E into a bit-vector annotation that contains νf_E as well.

Thus, **solve** correctly computes the greatest fixed point of f_E.

Theorem 3.1 (Correctness)
Let $\mathcal{T} = \langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$ be an extended labeled transition system, E be a closed set of simple equations, and $\nu f_E = \langle \nu_1, \ldots, \nu_n \rangle$ be the greatest fixed point of f_E. Then **solve** *terminates, and for any $s \in \mathcal{S}$ we have that $s \in \nu_i$ if and only if $s.X[i] =$* true.

Finally, we state and prove our complexity result.

Theorem 3.2 (Complexity)
Let $\mathcal{T} = \langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$ be an extended labeled transition system and E be a closed set of simple equations. Then the worst-case time complexity of **solve** *is $O(|\mathcal{T}| * |E|)$, where $|\mathcal{T}| = |\mathcal{S}| + |\rightarrow|$ and $|E|$ is the number of equations in E.*

Proof. In order to prove this theorem, it is enough to prove the result for **initialize** and **update**. The case of **initialize** it straightforward. Thus it remains to show that **update** takes time proportional to $|\mathcal{T}|*|E|$. The first thing to notice is that, in the worst-case, every state-variable pair will be added to M once and will require analysis. This is because the corresponding bit-vector component can only be set to *false* once during the analysis. Thus there exists a constant c so that the worst-case time complexity is bounded by the following expression:

$$\begin{array}{l} c * \sum_{\langle s, X_i \rangle} |\{ X_j \mid X_i \stackrel{\vee}{\rightarrow} X_j \vee X_i \stackrel{\wedge}{\rightarrow} X_j \}| + \\ c * \sum_{\langle s, X_i \rangle} |\{ s' \mid s' \stackrel{a}{\rightarrow} s \}| * |\{ X_j \mid X_i \stackrel{\langle a \rangle}{\rightarrow} X_j \vee X_i \stackrel{[a]}{\rightarrow} X_j \}| \end{array}$$

This fact relies on the observation that (the sets of pairs) $\stackrel{\vee}{\rightarrow}$, $\stackrel{\wedge}{\rightarrow}$, $\stackrel{\langle a \rangle}{\rightarrow}$ and $\stackrel{[a]}{\rightarrow}$ are pairwise-disjoint. Now this expression can be evaluated as follows to complete the proof of the theorem:

$$\begin{array}{rl} & c * \sum_{s} \sum_{X_i} |\{ X_j \mid X_i \stackrel{\vee}{\rightarrow} X_j \vee X_i \stackrel{\wedge}{\rightarrow} X_j \}| + \\ & c * \sum_{s} \sum_{X_i} |\{ s' \mid s' \stackrel{a}{\rightarrow} s \}| * |\{ X_j \mid X_i \stackrel{\langle a \rangle}{\rightarrow} X_j \vee X_i \stackrel{[a]}{\rightarrow} X_j \}| \\ \leq & c * (|\mathcal{S}| * |E| + \sum_{s} |\{ s' \mid s' \stackrel{a}{\rightarrow} s \}| * |E|) \\ = & c * (|\mathcal{S}| + |\rightarrow|) * |E| \\ = & c * |\mathcal{T}| * |E| \end{array}$$

□

4 Verifying Behavioural Relations

In this section we show how the model checking algorithm presented in the previous section may be applied to the efficient verification of behavioral relations between finite-state extended labeled transition systems. Central to our approach is the *prebisimulation preorder* $\sqsubseteq\!\!\!\!\sim$, which relates states (processes) in a given extended labeled transition system on the basis of their transitions and their relative "definedness" [Sti, Wa].

Definition 4.1 *Let $\langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$ be an extended labeled transition system. Then $\sqsubseteq\!\!\!\!\sim \subseteq \mathcal{S} \times \mathcal{S}$ is the largest relation satisfying the following, whenever $s_1 \sqsubseteq\!\!\!\!\sim s_2$:*

1. *$s_1 \stackrel{a}{\rightarrow} s_1'$ implies $s_2 \stackrel{a}{\rightarrow} s_2'$ and $s_1' \sqsubseteq\!\!\!\!\sim s_2'$ for some s_2'.*
2. *If $s_1 \downarrow a$ then:*
   - *(a) $s_2 \downarrow a$, and*
   - *(b) $s_2 \stackrel{a}{\rightarrow} s_2'$ implies $s_1 \stackrel{a}{\rightarrow} s_1'$ and $s_1' \sqsubseteq\!\!\!\!\sim s_2'$ for some s_1'.*

It is straightforward to establish that such a largest relation exists and is a preorder. Definition 4.1 may be extended to states from (state-disjoint) transition systems as follows. Let $\langle \mathcal{S}_1, Act_1, \rightarrow_1, \uparrow_1 \rangle$ and $\langle \mathcal{S}_2, Act_2, \rightarrow_2, \uparrow_2 \rangle$ be two such transition systems, and let $s_1 \in \mathcal{S}_1$ and $s_2 \in \mathcal{S}_2$. Then $s_1 \sqsubseteq\!\!\!\!\sim s_2$ is defined to hold if it holds in $\langle \mathcal{S}_1 \cup \mathcal{S}_2, Act_1 \cup Act_2, \rightarrow_1 \cup \rightarrow_2, \uparrow_2 \cup \uparrow_2 \rangle$.

In [CH, Ste, SI] it is shown how various other process equivalences and preorders, including observational equivalence [Mi1, Mi2], the divergence preorder [Wa], the testing preorder [DH], and trace containment, may be computed by applying appropriate transformations to the extended labeled transition systems and then determining the prebisimulation preorder. These relations have all been studied extensively as means for verifying processes; this stresses the importance of the prebisimulation preorder and motivates the need for efficient tools for its automatic verification. To our knowledge, the model-checking-based algorithm we are going to develop in the remainder of the paper is the most efficient algorithm for preorder checking that has been published until now.

4.1 Characteristic Formulae

Steffen [Ste] and Ingólfsdóttir [SI] have shown how one may construct a set E of propositional equations, one for each state, from an extended labeled transition system $\mathcal{T}$ that characterizes the prebisimulation preorder for states in $\mathcal{T}$ in the following sense. Let s be a state in $\mathcal{T}$. Then for any state s' in any extended labeled transition system, $s \sqsubseteq s'$ if and only if $s' \models_E X_s$, where X_s is the left-hand side of the equation associated with s in E. Here we briefly review their setup.

The logic used in [SI] is essentially a special case of the logic proposed in Section 2, where the atomic formulas are given by the subsets of Act (note that Act is finite, since we are restricting our attention to finite-state systems). The semantics of such a formula $A \subseteq Act$ is the following:

$$[\![A]\!]e \;=\; \{\, s \mid \{\, a \mid s \xrightarrow{a} \;\vee\; s{\uparrow}a \,\} \subseteq A \,\}$$

Thus a state s satisfies A if every action initially available to s, or on which s may diverge, is contained in A.

For an extended labeled transition system $\langle \{s_1, \ldots, s_n\}, Act, \rightarrow, \uparrow \rangle$ the characteristic set of equations E is now given by:

$$X_i \;=\; \bigwedge_{\{\langle a,s_j\rangle \mid s_i \xrightarrow{a} s_j\}} \langle a \rangle X_j \;\wedge\; \bigwedge_{\{a \mid s{\downarrow}a \wedge s \xrightarrow{a}\}} [a]\Big(\bigvee_{\{s_j \mid s_i \xrightarrow{a} s_j\}} X_j \Big) \;\wedge\; \{\, a \mid s_i \xrightarrow{a} \vee s_i{\uparrow}a \,\}$$

where X_i is the variable associated to s_i for every states s_i of the transition system. The main theorem in [SI] is the following.

Theorem 4.2 *Let $\mathcal{T}$ be an extended labeled transition system and s one of its states. Furthermore, let E be the characteristic equation set of $\mathcal{T}$ and X_s the variable in E associated with s. Then for any state s' in any extended labeled transition system, $s \sqsubseteq s'$ if and only if $s' \models_E X_s$.*

4.2 Computing the Prebisimulation Preorder

The result from the previous section suggests a technique for computing the prebisimulation preorder between states in transition systems: construct the characteristic equation set, apply the model-checking procedure from Section 3, and determine whether the desired state is in the set associated with the relevant variable. To calculate the complexity of this approach, we note the following.

- Given an extended labeled transition system $\mathcal{T}$, the characteristic equation set E may be built in $O(|\mathcal{T}|)$ time. Moreover, $|E|$ is $O(|\mathcal{T}|)$.
- A simple equation set for $|E|$ may be constructed in $O(|E|)$ time.
- Model checking of an extended labeled transition system $\mathcal{T}'$ with respect to E can be done in time proportional to $|\mathcal{T}'| * |E|$.

Therefore, computing whether $s \underset{\sim}{\sqsubset} s'$, where s and s' are states of $\mathcal{T}$ and $\mathcal{T}'$ respectively, may be performed in $O(|\mathcal{T}| * |\mathcal{T}'|)$ time. This improves dramatically on the complexity result mentioned in [CPS1, CPS2], which is $O(\,(|\mathcal{S}| + |\mathcal{S}'|)^4 \; * \; (|\mathcal{T}| + |\mathcal{T}'|)\,)$, in the following sense. For a fixed action set, the new result is of fourth order in the number of states of the transition systems involved, whereas the previous algorithm exhibits complexity that is sixth order in its worst case.

5 Conclusions and Future Work

In this paper we have presented a linear-time algorithm for model checking in a variant of the modal mu-calculus, and we have shown how it may be used to compute the prebisimulation preorder. Both algorithms are more efficient than existing algorithms in the literature. As the prebisimulation preorder may be used as the basis for defining many behavioral preorders and equivalences on processes, our approach covers these relations as well.

There are a number of directions that may pursued from this work. First, it is a straightforward matter to modify our algorithm to compute least fixed points as well as greatest fixed points. Provided that *alternated fixed points* are excluded [EL], i.e. there are no mutually recursive greatest and least fixed point equations, the resulting algorithm would have the same complexity as the one presented in this paper. Once least fixed points are added in this fashion, the formulas of the temporal logic CTL [CES] may be translated via a linear-time procedure into formulas in our logic. As a result, the CTL model checking algorithm obtained by applying our more general procedure to the translated formulas would have the same complexity as the model checking procedure described in [CES].

It would also be interesting to see how the algorithm in this paper could be extended to handle the full modal mu-calculus, which includes alternating fixed points [Ko]. Algorithms of this generality can be found in [C, EL, SW, Wi]. However, only Emerson and Lei [EL] give a complexity analysis. Their algorithm is exponential in $ad + 1$, where ad, the *alternation depth* of the formula, is a measure of the degree of mutual recursion among greatest and least fixed points.

Our algorithm outperforms this algorithm in the special case that the alternation depth is 1 (i.e. there are no mutually recursive greatest and least fixed points): our approach is linear, while theirs is quadratic. We conjecture that our algorithm can be generalized into an algorithm for the entire mu-calculus that is exponential just in ad.

Finally, our algorithm may be used to automate efficiently the *compositional* proof techniques in [CS, GS], which rely on determining whether processes have certain safety properties that can be expressed in the logic studied here. We also plan to implement this algorithm as an extension of the Concurrency Workbench [CPS1, CPS2].

References

[AC] Arnold, A., and P. Crubille. "A Linear Algorithm To Solve Fixed-Point Equations on Transition Systems." *Information Processing Letters*, v. 29, 30 September 1988, pp. 57–66.

[BSV] Boudol, G., de Simone, R. and Vergamini, D. "Experiment with Auto and Autograph on a Simple Case Sliding Window Protocol." INRIA Report 870, July 1988.

[CES] Clarke, E.M., E.A. Emerson and Sistla, A.P. "Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications." *ACM Transactions on Programming Languages and Systems*, v. 8, n. 2, 1986, pp. 244-263.

[C] Cleaveland, R. "Tableau-Based Model Checking in the Propositional Mu-Calculus." *Acta Informatica*, 1990.

[CH] Cleaveland, R. and Hennessy, M.C.B. "Testing Equivalence as a Bisimulation Equivalence." In *Proceedings of the Workshop on Automatic Verification Methods for Finite-State Systems*. Lecture Notes in Computer Science series 407, Springer-Verlag, Berlin, 1989.

[CPS1] Cleaveland, R., Parrow, J. and Steffen, B. "The Concurrency Workbench." In *Proceedings of the Workshop on Automatic Verification Methods for Finite-State Systems*, 1989, Lecture Notes in Computer Science 407, pp. 24–37. Springer-Verlag, Berlin.

[CPS2] Cleaveland, R., Parrow, J. and B. Steffen. *A Semantics based Verification Tool for Finite State Systems*, In pro. of the Ninth International Symposium on Protocol Specification, Testing, and Verification; North Holland, 1989.

[CS] Cleaveland, R. and Steffen, B. "When is 'Partial' Complete? A Logic-Based Proof Technique using Partial Specifications." In Proceedings LICS'90, 1990.

[DH] DeNicola, R. and Hennessy, M.C.B. "Testing Equivalences for Processes." *Theoretical Computer Science* 24, 1984, pp. 83-113.

[EL] Emerson, E.A. and Lei, C.-L. "Efficient Model Checking in Fragments of the Propositional Mu-Calculus." In *Proceedings of the First Annual Symposium on Logic in Computer Science*, 1986, pp. 267-278.

[Fe] Fernandez, J.-C. *Aldébaran: Une Système de Vérification par Réduction de Processus Communicants*. Ph.D. Thesis, Université de Grenoble, 1988.

[GS] Graf, S. and Steffen, B. "Using Interface Specifications for Compositional Reduction." To appear in *Proceedings of the Workshop on Computer-Aided Verification.*

[Ko] Kozen, D. "Results on the Propositional μ-Calculus." *Theoretical Computer Science*, v. 27, 1983, pp. 333-354.

[La] Larsen, K.G. "Proof Systems for Hennessy-Milner Logic with Recursion." In *Proceedings of CAAP*, 1988.

[MSGS] Malhotra, J., Smolka, S.A., Giacalone, A. and Shapiro, R. "Winston: A Tool for Hierarchical Design and Simulation of Concurrent Systems." In *Proceedings of the Workshop on Specification and Verification of Concurrent Systems*, University of Stirling, Scotland, 1988.

[Mi1] Milner, R. *A Calculus of Communicating Systems.* Lecture Notes in Computer Science 92. Springer-Verlag, Berlin, 1980.

[Mi2] Milner, R. *Communication and Concurrency*, Prentice Hall, 1989.

[PS] Plotkin, G. and Stirling, C. "A Framework for Intuitionistic Modal Logics." Theoretical Aspects of Reasoning about Knowledge, Monterey, 1986.

[RRSV] Richier, J., Rodriguez, C., Sifakis, J. and Voiron, J. "Verification in XESAR of the Sliding Window Protocol." In *Proceedings of the Seventh IFIP Symposium on Protocol Specification, Testing, and Verification*, 1987, North-Holland.

[Ste] Steffen, B.U. "Characteristic Formulae for CCS with Divergence." In *Proceedings ICALP*, Lecture Notes in Computer Science 372, pp. 723-733. Springer-Verlag, Berlin, 1989.

[SI] Steffen, B.U., and Ingólfsdóttir, A. "Characteristic Formulae for CCS with Divergence." To appear in *Theoretical Computer Science*.

[Sti] Stirling, C. "Modal Logics for Communicating Systems." *Theoretical Computer Science*, v. 49, 1987, pp. 311-347.

[SW] Stirling, C., and Walker, D. "Local Model Checking in the Modal Mu-Calculus." In Proceedings CAAP'89, Lecture Notes in Computer Science 351, pp. 369 - 383, 1989.

[Ta] Tarski, A. "A Lattice-Theoretical Fixpoint Theorem and its Applications." *Pacific Journal of Mathematics*, v. 5, 1955.

[Wa] Walker, D. "Bisimulations and Divergence." In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, 1988, pp. 186-192. Computer Society Press, Washington DC.

[Wi] Winskel, G. "On the Compositional Checking of Validity." In Proceedings CONCUR'90, Lecture Notes in Computer Science 458, pp. 481 - 501, 1990.

Appendix: The Algorithms

The Initialization Algorithm

Input:

- An extended labeled transition system $\mathcal{T} = \langle \mathcal{S}, Act, \rightarrow, \uparrow \rangle$, whose states s are annotated with the atomic propositions that hold of s.
- An auxiliary graph G whose nodes are the variables of the equation system E and whose edges describe their mutual dependencies.

Output:

- An extended labeled transition system whose states are annotated with appropriately initialized bit-vectors $s.X$ and counters $s.C$.
- A list M, initialized with pairs $\langle s, X_i \rangle$ that determine the reinvestigation process.

```
procedure initialize ((S, Act, →, ↑), G);

  M := empty;
  for each s ∈ S do
    for i := 1 to n do
    begin
      case right hand side of X_i in E is:

           A:          s.X[i] := s.A

           X_j ∨ X_k:  begin s.C[i] := 2; s.X[i] := true end

           ⟨a⟩X_j:     begin
                         s.C[i] := |{ s' | s -a-> s' }|;
                         if s.C[i] = 0
                         then s.X[i] := false
                         else s.X[i] := true
                       end

           [a]X_j:     if s↑a
                       then s.X[i] := false
                       else s.X[i] := true

           ELSE:       s.X[i] := true;

      if not(s.X[i]) then add ⟨s, X_i⟩ to M
    end
end.
```

The Update Procedure

Input:

- An extended labeled transition system whose states are annotated with appropriately initialized bit-vectors $s.X$ and counters $s.C$.
- A list M, initialized with pairs $\langle s, X_i\rangle$ that determine the reinvestigation process.
- An auxiliary graph G whose nodes are the variables of the equation system E and whose edges describe their mutual dependencies.

Output:

- An extended labeled transition system whose states are annotated with bit-vectors $s.X$ that represent the greatest fixed point νf_E.

procedure update $((\mathcal{S}, Act, \rightarrow, \uparrow), M, G)$;

```
while M is not empty do begin
  remove the first pair from M, calling it ⟨s, X_i⟩;

  foreach X_j such that X_i -∨-> X_j do begin
    s.C[j] := s.C[j] − 1;
    if s.C[j] = 0 then begin
      s.X[j] := false;
      add ⟨s, X_j⟩ to M
    end
  end;

  foreach X_j such that X_i -∧-> X_j do
    if s.X[j] then begin
      s.X[j] := false;
      add ⟨s, X_j⟩ to M
    end;

  foreach X_j such that X_i -⟨a⟩-> X_j do
    foreach s' such that s' -a-> s do begin
      s'.C[j] := s'.C[j] − 1;
      if s'.C[j] := 0 then begin
        s'.X[j] := false;
        add ⟨s', X_j⟩ to M
      end
    end;

  foreach X_j such that X_i -[a]-> X_j do
    foreach s' such that s' -a-> s and s'.X[j] do begin
      s'.X[j] := false;
      add ⟨s', X_j⟩ to M
```

```
      end
   end
end.
```